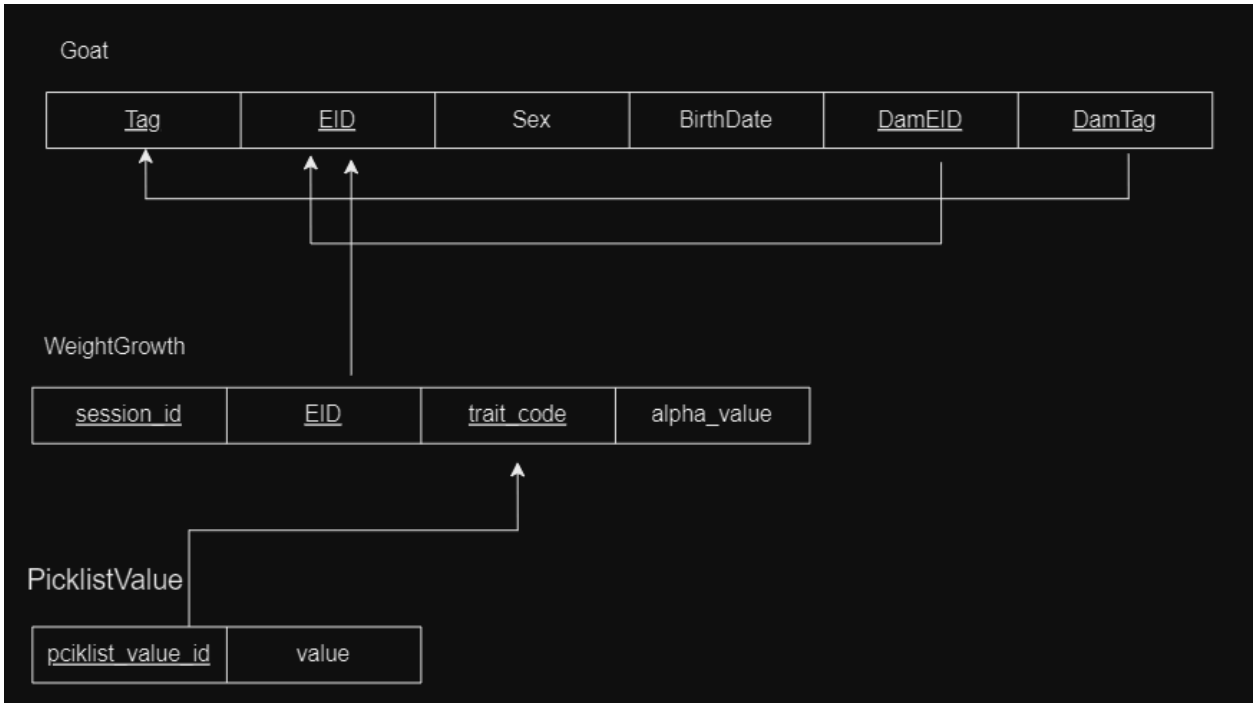
The above table is not in 1NF because LiveWeight isn't an atomic attribute as it would be a list of all the values associated with dates that the goat was measured.

To fix this, Live Weight needs to be placed into another table



The database already has one of these tables, which we will only use to determine weight growth. It also has a weak entity, PicklistValue that determines the type of trait it is. This will be used to only select the weight measurements.
These tables are all in 1NF now that there are no multivalued attributes and each attribute is atomic.
Additionally, it is in 2NF because each non prime attribute is fully functionally on the entire primary key
Because PicklistValue is a separate table, it is in 3NF. If it was combined with WeightGrowth, it would be transitively dependent on trait_code. We solve this by placing it in a new relation.
Lastly, it doesn't violate BCNF because Goat has superkey {Tag, EID}, WeightGrowth has the superkey {session_id}, and PicklistValue has the superkey {picklist_value_id}. Every attribute is dependent on a prime attribute.

VIEWS

Birth Cohort

- Tag
- EID
- BirthDate
- Must allow the user to group by a range of birth dates

Family

- Tag
- DamTag
- EID
- DamEID
- By choosing a single child dam,

Total Herd

- Tag
- EID

Birth Cohort View

```
CREATE OR REPLACE VIEW GoatBirthCohort AS
        SELECT Tag, EID, BirthDate
        FROM goat
        WHERE BirthDate > '[lower bound date]' AND BirthDate < '[upper bound date]';
```

Family View

```
CREATE OR REPLACE VIEW GoatFamilyTree AS
WITH RECURSIVE FamilyTree AS (
        SELECT Tag, EID, BirthDate, DamTag, DamEID
        FROM Goat
        WHERE Tag = [insert mother tag] AND EID = [insert mother eid]

        UNION ALL

        SELECT g.Tag, g.EID, g.BirthDate, g.DamTag, g.DamEID
        FROM goat g
        INNER JOIN FamilyTree ft on g.DamTag = ft.Tag AND g.DamEID = ft.EID
)
SELECT * FROM FamilyTree;
```

Total Herd View

```
CREATE OR REPLACE VIEW TotalHerd AS
```

```
SELECT Tag, EID
FROM goat;
```

Explanation:
The above view, Birth Cohort View, helps to support our use case as we will be able to select two dates that will define a birth cohort, and the view will return all goats that were born within that range. We can then use that information to calculate the average growth rate of that birth cohort.

The view Total Herd View will also be used in a similar fashion. It will return all the goats in the database and calculate the given average growth rate.

Lastly, the view Family View will allow the user to input the primary key for any dam, and recursively find all related goats and return them. This will allow us to calculate the average growth rate of all goats related to that dam and determine any significant differences between genetics and growth rate.